Exercises for Transaction Systems, summer term 2016
Linnea Passing (linnea.passing@tum.de)
http://www-db.in.tum.de/teaching/ss16/transactions/

Sheet No. 11 – Sample solution

Exercise 1 (6 points) Show for the IDM model of transactions:
(a) Final state serializability is not monotone.
(b) Conflict serializability is monotone.

Solution (a) Proof by example: Take the example from chapter 8 slide 14: $d_1(3) d_2(3) m_1(1; 2) m_2(2; 3) m_1(3; 4)$ and add a third transaction at the end that empties the whole database. Now $t_1 t_2 t_3$ and $t_2 t_1 t_3$ are FSR-equivalent serial schedules, but projections to $t_1 t_2$ cannot be made serial.
(b) Removing transactions only removes edges from the graph. Removing edges from an acyclic graph cannot create cycles.

Exercise 2 (4 points) Discuss how predicate locking can be extended to disjunctive conditions such as queries of the form
\[
\text{select name from employees where position='Manager' or department='Research'}; \\
\text{also discuss how join queries such as}
\]
\[
\text{select e.name, e.department from employees e, department d}
\text{where e.position='Manager' and d.city='Toronto' and e.department=d.department};
\]
could be (conservatively) handled by predicate locking.

Solution Disjunctive: treat them as two separate queries, find a shape that covers all disjunctions, extend the system to store multiple C per query and link them together so that the system knows it has to check them together
Join: ignore the join and lock all “input” tuples to the join (i.e. apply all predicates that span only one table, lock the Cartesian product)

Exercise 3 (10 points) Consider the following B+-tree index on the attribute accountnumber of an accounts table. Assume that all tree nodes have a space capacity for holding up to four entries. Write down all locking and unlocking operations necessary for the execution of the following transaction, assuming incremental key range locking at the access layer and lock coupling at the page layer.
\[
\begin{align*}
& \text{begin transaction;} \\
& \text{select count(*) from accounts where accountnumber between 11 and 25;} \\
& \text{insert into accounts (accountnumber, ...) values (27, ...);} \\
& \text{commit transaction;}
\end{align*}
\]
Solution The transaction translates to: `range_search(11, 25)` and `insert(27, ...)`

```
range_search(11, 25):
search(11) ret 12 q3 :
lock r, lock p1, unlock r, lock q3, unlock p1, lock q2, unlock q3, lock9, unlock q2
next(12, q3, 25) ret 13 q3 : lock 13

    ... 

next(24, q5, 25) ret nil nil : lock 24
(there are nine next steps)

insert(27, ...):
lock r, lock p2, unlock r, lock q6, unlock p2, lock q5, lock 24, lock 27, unlock q5, unlock q6
```

Do not use locks for the next steps, the next pointers in the leaf nodes of the B+-tree are sufficient to find all values in the search range.